

10/ 630176

cog C



**PATENT**  
Attorney Docket No. 49769

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

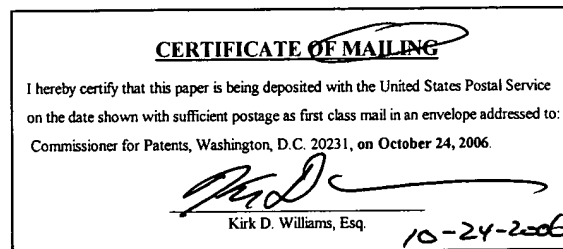
Patent No. 7,082,492

Confirmation No. 7655

Issued: July 25, 2006

Name of Patentee: Pullela et al.

Patent Title: ASSOCIATIVE MEMORY  
ENTRIES WITH FORCE NO HIT AND  
PRIORITY INDICATIONS OF PARTICULAR  
USE IN IMPLEMENTING POLICY MAPS  
IN COMMUNICATION DEVICES



**REQUEST FOR CERTIFICATE OF CORRECTION OF  
PATENT FOR PATENT OFFICE MISTAKE (37 C.F.R. § 1.322)**

Attn: Certificate of Correction Branch  
Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

Dear Sir:

It is requested that a Certificate of Correction be issued to correct Office mistakes found the above-identified patent. Attached hereto is a Certificate of Correction which indicates the requested correction. For your convenience, also attached are copies of selected pages (a) from the issued patent with errors highlighted, and (b) from the original application as filed July 29, 2003 with the correct text/instructions.

**Certificate  
NOV 02 2006  
of Correction**

**NOV 03 2006**


In re US Patent No. 7,082,492

It is believed that there is no charge for this request because applicant or applicants were not responsible for such error, as will be apparent upon a comparison of the issued patent with the application as filed or amended. However, the Assistant Commissioner is hereby authorized to charge any fee that may be required to Deposit Account No. 501430.

Respectfully submitted,  
**The Law Office of Kirk D. Williams**

Date: October 24, 2006

By

 10-24-2006  
Kirk D. Williams, Reg. No. 42,229  
One of the Attorneys for Applicants  
CUSTOMER NUMBER 26327  
The Law Office of Kirk D. Williams  
1234 S. OGDEN ST., Denver, CO 80210  
303-282-0151 (telephone), 303-778-0748 (facsimile)

**UNITED STATES PATENT AND TRADEMARK OFFICE**  
**CERTIFICATE OF CORRECTION**

PATENT NO. : 7,082,492

DATED : July 25, 2006

INVENTOR(S) : Pullela et al.

It is certified that error(s) appear in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Col. 8, line 39, replace "FIGS 1A-E;" with -- FIGS 1A-E --

Col. 17, line 11, replace "profile I) 521" with -- profile ID 521 --

Col. 23, line 26, replace "9601" with -- 960I --

**MAILING ADDRESS OF SENDER:**

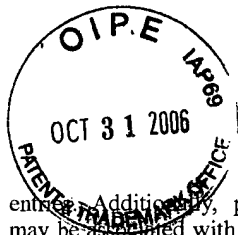
Kirk D. Williams, Reg. No. 42,229  
Customer No. 26327  
The Law Office of Kirk D. Williams  
P.O. Box 61538, Denver, CO 80206

PATENT NO. 7,082,492

No. of additional copies

⇒ NONE (0)

05 2006



entries. Additionally, programmable priority indications may be associated with one or more of the entries, or with the associative memory devices, associative memory banks, etc. The force no-hit indications are often used in response to identified deny instructions in an access control list or other policy map. A lookup operation is then performed on these associative memory entries, with highest matching result or results identified based on the programmed and/or implicit priority level associated with the entries, or with the associative memory devices, associative memory banks, etc.

Methods and apparatus are disclosed for performing lookup operations using associative memories, including, but not limited to modifying search keys within an associative memory based on modification mappings, forcing a no-hit condition in response to a highest-priority matching entry including a force no-hit indication, selecting among various associative memory blocks or sets or banks of associative memory entries in determining a lookup result, and detecting and propagating error conditions. In one embodiment, each block retrieves a modification mapping from a local memory and modifies a received search key based on the mapping and received modification data. In one embodiment, each of the associative memory entries includes a field for indicating that a successful match on the entry should or should not force a no-hit result. In one embodiment, an indication of which associative memory sets or banks or entries to use in a particular lookup operation is retrieved from a memory.

One embodiment performs error detection and handling by identifying, handling and communication errors, which may include, but is not limited to array parity errors in associative memory entries and communications errors such as protocol errors and interface errors on input ports. Array parity errors can occur as a result of failure-in-time errors which are typical of semiconductor devices. One embodiment includes a mechanism to scan associative memory entries in background, and to identify any detected errors back to a control processor for re-writing or updating the flawed entry. In one embodiment, certain identified errors or received error conditions are of a fatal nature in which no processing should be performed. For example, in one embodiment, a fatal error causes an abort condition. In response, the device stops an in-progress lookup operation and just forwards error and possibly no-hit signals. Typically, these signals are generated at the time the in-progress lookup operation would have generated its result had it not been aborted so as to maintain timing among devices in a system including the associative memory.

In one embodiment, including cascaded or connected associative memory devices, error status messages indicating any error type and its corresponding source are propagated to indicate the error status to the next device and/or a control processor. In addition, the communicated signal may indicate and generate an abort condition in the receiving device. In one embodiment, the receiving device does not perform its next operation or the received instruction, or it may abort its current operation or instruction. Moreover, the receiving device may or may not delay a time amount corresponding to that which its processing would have required in performing or completing the operation or instruction so as to possibly maintain the timing of a transactional sequence of operations.

One embodiment generates accounting or other data based on that indicated in an access control list or other specification, and typically using associative memory entries in one or more associative memory banks and/or memory devices. One embodiment identifies an access control list

including multiple access control list entries, with a subset of these access control list entries identifying accounting requests. Accounting mechanisms, such as, but not limited to counters or data structures, are associated with each of said access control list entries in the subset of access control list entries identifying accounting requests. An item is identified. A particular one of the accounting mechanisms corresponding to the item is identified and updated. In one embodiment, the item corresponds to one or more fields of a received packet. In one embodiment, the item includes at least one autonomous system number, said at least one autonomous system number identify a set of communication devices under a single administrative authority. In one embodiment, at least one of the accounting mechanisms is associated with at least two different access control list entries in the subset of access control list entries identifying accounting requests.

One embodiment merges lookup results, such as from one or more associative memory banks and/or memory devices.

One embodiment identifies an access control list including multiple access control list entries. A first set of access control list entries corresponding to a first feature of the access control list entries and a second set of access control list entries corresponding to a second feature of the access control list entries are identified. A first associative memory bank is programmed with the first associative memory entries and a second associative memory bank is programmed with the second associative memory entries, with the first associative memory entries having a higher lookup precedence than the second associative memory entries. A lookup value is then identified, such as that based on a packet or other item. Lookup operations are then typically performed substantially simultaneously on the first and second sets of associative memory entries to generate multiple lookup results, with these results typically being identified directly, or via a lookup operation in an adjunct memory or other storage mechanism. These lookup results are then combined to generate a merged lookup result.

FIGS. 1A-E; are block diagrams of various exemplary systems and configurations thereof, with these exemplary systems including one or more embodiments for performing lookup operations using associative memories. First, FIG. 1 illustrates one embodiment of a system, which may be part of a router or other communications or computer system, for performing lookup operations to produce results which can be used in the processing of packets. In one embodiment, control logic 110, via signals 111, programs and updates associative memory or memories 115, such as, but not limited to one or more associative memory devices, banks, and/or sets of associative memory entries which may or may not be part of the same associative memory device and/or bank. In one embodiment, control logic 110 also programs memory 120 via signals 123. In one embodiment, control logic 110 includes custom circuitry, such as, but not limited to discrete circuitry, ASICs, memory devices, processors, etc.

In one embodiment, packets 101 are received by packet processor 105. In addition to other operations (e.g., packet routing, security, etc.), packet processor 105 typically generates one or more items, including, but not limited to one or more packet flow identifiers based on one or more fields of one or more of the received packets 101 and possibly from information stored in data structures or acquired from other sources. Packet processor 105 typically generates a lookup value 103 which is provided to control logic 110 for providing control and data information (e.g., lookup words, modification data, profile IDs, etc.) to associative memory or

no semicolon

17

FIG. 5A illustrates of an output selector 500 (which may or may not correspond to an output selector 231–232 of FIG. 2) used in one embodiment. As shown, output selector 500 includes control logic 510 and memory 511. In one embodiment, programming signals 504 are received, and in response, one or more data structures in memory 511 are updated.

FIG. 5B illustrates one data structure used in one embodiment. Available array 520 is programmed with an associative memory blocks and optionally previous stage results available for use indicator 525 for each (profile ID) 521 to be used. Each indicator 525 identifies which, if any, associative memory blocks, sets of entries or associative memory banks are to be considered in determining which matching associative entry to select for the ultimate highest-priority matching associative memory entry. In one embodiment, indicator 525 further identifies which previous stage results to consider. In one embodiment, a priority level is associated with each of the banks and/or previous stage results. Thus, based on a profile ID 521 received over via selector control signal 501 (FIG. 5A), available array 520 can be retrieved from memory 511 (FIG. 5A). In one embodiment, there is an implied priority ordering of associative memory blocks and any previous stage results, while in one embodiment this priority ordering for determining the ultimate highest-priority matching entry is programmable and/or variable per lookup operation. In one embodiment, associative memory blocks available for use indicator 525 is a bitmap data structure, while in one embodiment, associative memory blocks available for use indicator 525 is a list, set, array, or any other data structure.

Returning to FIG. 5A, in the performance of a lookup operation, output selector 500 receives selector control signal 501, which may include a profile ID. In addition, output selector 500 receives any relevant previous stage results 502 and results 503 from zero or more of the associative memory blocks from which the highest-priority entry will be selected, and which, if any, will be identified in generated result 515.

Moreover, in one embodiment, selector control signal 501 including an enable indication, the enable indication including an enabled or not enabled value, such that in when a not enable value is received, output selector 500 is not enabled and does not select among results from blocks 1–N 503 or optional previous stage results 502. In one embodiment, when not enabled, output selector 500 generates a result signal 515 indicating a no hit, not enabled, or some other predetermined or floating value.

Additionally, in one embodiment, result 515 is communicated over a fixed output bus, which may or may not be multiplexed with other results 515 generated by other output selectors 500. In one embodiment, the associative memory may include one or more output buses, each typically connected to a single pin of a chip of the associative memory, with the selection of a particular output bus possibly being hardwired or configurable, with the configuration possibly being on a per lookup basis, such as that determined from a received value or configuration information retrieved from a memory (e.g., based on the current profile ID.) In such a configuration, control logic 510 (or other mechanism) typically selects which output bus (and the timing of sending result 515) to use for a particular or all results 515.

A process used in one embodiment for receiving and selecting a highest-priority associative memory entry, if any, is illustrated in FIG. 5C. Processing begins with process block 540, and proceeds to process block 542, wherein the

18

results from the associative memory blocks and the profile ID are received. In process block 544, the set of associative memory blocks to consider in determining the result is retrieved from a data structure/memory based on the profile ID. In process block 546, any relevant previous stage results are received from coupled associative memories. Next, in process block 548, the highest priority match from the available associative memory block and previous stage results is identified, if any, based on the implied and/or programmed priority values associated with the matching entries and/or associative memories, blocks, etc. Then, in process block 550, the result is communicated over a fixed or identified output bus/pin or to some other destination, with the result typically including a no hit indication or a hit indication and an identification of the ultimate highest-priority matching associative memory entry. Processing is complete as indicated by process block 552.

FIG. 6A illustrates an exemplary policy map 600, including deny and permit instructions. Note, there are many applications of embodiments, and not all use permit and deny instructions. FIG. 6B illustrates associative memory entries 621 and 622 as determined by one embodiment based on policy map 600. Associative memory entries 621 and 622 could be programmed in a same or different associative memories or associative memory blocks. Associative memory entries 621 and 622 are shown in separate groupings to illustrate how priority can be optionally used and programmed in one embodiment. As shown, the deny statements in policy map 600 generate force no-hit indications (e.g., FORCE NO-HIT=1) in corresponding entries of entries 621 and 622.

By using the optional priority indications, entries 621 and 622 can be stored in different associative memories and/or associative memory banks, etc., to possibly consider in determining where to store the entries in order to efficiently use the space available for the entries. By associating a priority level with each entry, entries within a same associative memory and/or associative memory block, etc. can have different priority levels, which gives great flexibility in programming and managing the entries and space available for storing the entries.

FIG. 6C illustrates a data structure 650 for indicating priority of associative memories, blocks, or entries, etc. used in one embodiment. As shown, priority mapping data structure 650 provides a priority indication 652 (e.g., value) for each of the associative memories, associative memory blocks, associative memory entries, etc. (identified by indices 651). Associative memories and/or blocks, etc. associated with programmed priority values can be used with or without programmed priority values associated with the associative memory entries themselves.

FIG. 7A illustrates a process for programming associative memory entries used in one embodiment. Processing begins with process block 700, and proceeds to process block 702, wherein a policy map (e.g., any definition of desired actions, etc.) is identified. Next, in process block 704, a set of corresponding entries is identified based on the policy map. In process block 706, a force no-hit indication is associated with one or more of the entries (if so correspondingly defined by the policy map). A force no-hit indication is of particular use in implementing deny operations, but is not required to be identified with a deny operation. Next, in process block 708, optionally, priority indications are associated with each of the entries, associative memories, associative memory banks, etc. In process block 710, one or more associative memories and/or banks are programmed

profile ID  
≡

list entries. In process block 944, a second set of the access control list entries corresponding to a second feature of the access control list entries is identified. In process block 945, a second associative memory bank and a second adjunct memory are programmed with entries corresponding to the second set of access control list entries. The first set of associative memory entries have a higher lookup precedence than the second set of associative memory entries. Processing is complete as indicated by process block 946.

FIG. 9F illustrates a process used by one embodiment to perform lookup operations and to identify the merged result. Processing begins with process block 950, and proceeds to process block 951, wherein a lookup value is identified. Next, in process block 952, lookup operations are performed in the first and second associative memory banks and adjunct memories to generate first and second lookup results, which are merged in process block 953 to identify the merged result. Processing is complete as indicated by process block 954.

FIG. 9G illustrates a lookup value 960, result value 965, and merged result value 967 used in one embodiment. As shown, lookup value 960 includes a lookup type 960A, source address 960B, destination address 960C, source port 960D, destination port 960E, protocol type 960F, source ASN 960G, destination ASN 960H, and possibly other fields 960I. One embodiment uses all, less than all, or none of fields 960A-960I.

As shown, result value 965 includes a result type 965A, an action or counter indication 965B, and a precedence indication 965C. In one embodiment, result value 965 is programmed in the adjunct memories. One embodiment uses all, less than all, or none of fields 965A-965C.

As shown, merged result value 967 includes a result type 967A and an action or counter indication 967B. One embodiment uses all, less than all, or none of fields 967A-967B.

FIGS. 9H-9J illustrate merging logic truth tables 970, 972, and 974 for generating the merged result. In one embodiment, the merge result of a security lookup operation is illustrated in security combiner logic 970, and is based on the results of up to four substantially simultaneous (or not) lookup operations with differing precedence indicated in columns 970A-970D, with the corresponding merged result shown in column 970E. Note, the "..." in the fields indicate a don't care condition as a merged result corresponding to a higher priority will be selected.

In one embodiment, the merge result of a Quality of Service (QoS) lookup operation is illustrated in security combiner logic 972, and is based on the results of a previously merged security lookup operation and up to four substantially simultaneous (or not) lookup operations with differing precedence indicated in columns 972A-970E, with the corresponding merged result shown in column 972F.

In one embodiment, the merge result of an accounting lookup operation is illustrated in accounting combiner logic 972, and is based on the results of a previously merged security lookup operation and up to four substantially simultaneous (or not) lookup operations with differing precedence indicated in columns 974A-974E, with the corresponding merged result shown possibly identifying a counter to be updated in column 972F.

FIG. 9K illustrates a process used in one embodiment, to generate a security merged result, a QoS merged result, and an accounting merged result. Processing begins with process block 980, and proceeds to process block 981, wherein a packet is identified. Next, in process block 982, one or more FIB lookup operations are performed to identify source and

destination ASNs. In process block 983, a security lookup value is identified. In process block 984, lookup operations are performed based on the security lookup value in multiple associative memory banks and one or more adjunct memories to identify multiple security results, which are merged in process block 985 to identify the merged security result. Also, this merged security result is stored in a data structure or other mechanism for use in identifying the merged QoS and accounting results.

In process block 986, the QoS lookup value is identified. In process block 987, lookup operations are performed based on the QoS lookup value in multiple associative memory banks and one or more adjunct memories to identify multiple QoS results, which, in process block 988, are merged along with the previously determined merged security result to identify the merged QoS result.

In process block 989, the accounting lookup value is identified. In process block 990, lookup operations are performed based on the accounting lookup value in multiple associative memory banks and one or more adjunct memories to identify multiple accounting results, which, in process block 991, are merged along with the previously determined merged security result to identify the merged accounting result. Also, an identified counter or other accounting mechanism is updated. Processing is complete as indicated by process block 992.

In view of the many possible embodiments to which the principles of our invention may be applied, it will be appreciated that the embodiments and aspects thereof described herein with respect to the drawings/figures are only illustrative and should not be taken as limiting the scope of the invention. For example and as would be apparent to one skilled in the art, many of the process block operations can be re-ordered to be performed before, after, or substantially concurrent with other operations. Also, many different forms of data structures could be used in various embodiments. The invention as described herein contemplates all such embodiments as may come within the scope of the following claims and equivalents thereof.

What is claimed is:

1. A method for performing operations for programming one or more associative memories, the method comprising: identifying a specified policy map; determining a set of entries based on the specified policy map; and associating a force no-hit indication with one or more entries of the set of entries; wherein the force no-hit indication, when associated with a determined highest-matching entry of a group of entries participating in a lookup operation, causes the result of the lookup operation for the group of entries to be considered as not resulting in a hit.
2. The method of claim 1, comprising programming one or more associative memories with the set of entries.
3. The method of claim 1, comprising programming a plurality of banks of an associative memory with the set of entries.
4. The method of claim 3, comprising associating a priority indication with each entry of the set of entries.
5. The method of claim 4, comprising: programming a plurality of banks of an associative memory with the set of entries; and associating a programmable priority level with each of the plurality of banks.
6. The method of claim 1, wherein at least one of said one or more entries corresponds to a deny operation.

960I

associative memory entries to generate multiple lookup results, with these results typically being identified directly, or via a lookup operation in an adjunct memory or other storage mechanism. These lookup results are then combined to generate a merged lookup result.

6/8/12u39 5 **FIGS. 1A-E** are block diagrams of various exemplary systems and configurations thereof, with these exemplary systems including one or more embodiments for performing lookup operations using associative memories. First, FIG. 1 illustrates one embodiment of a system, which may be part of a router or other communications or computer system, for performing lookup operations to produce results which can be used in the processing of packets. In one embodiment, control logic 110, via signals 111, programs and updates associative memory or memories 115, such as, but not limited to one or more associative memory devices, banks, and/or sets of associative memory entries which may or may not be part of the same associative memory device and/or bank. In one embodiment, control logic 110 also programs memory 120 via signals 123. In one embodiment, control logic 110 includes custom circuitry, such as, but not limited to discrete circuitry, ASICs, memory devices, processors, etc.

In one embodiment, packets 101 are received by packet processor 105. In addition to other operations (e.g., packet routing, security, etc.), packet processor 105 typically generates one or more items, including, but not limited to one or more packet flow identifiers based on one or more fields of one or more of the received packets 101 and possibly from information stored in data structures or acquired from other sources. Packet processor 105 typically generates a lookup value 103 which is provided to control logic 110 for providing control and data information (e.g., lookup words, modification data, profile IDs, etc.) to associative memory or memories 115, which perform lookup operations and generate one or more results 117. In one embodiment, a result 117 is used is by memory 120 to produce a result 125. Control logic 110 then relays result 107, based on result 117 and/or result 125, to packet processor 105. In response, one or more of the

indication 420 (FIG. 4C) is used. Processing is complete as indicated by process block 499.

FIG. 5A illustrates of an output selector 500 (which may or may not correspond to an output selector 231-232 of FIG. 2) used in one embodiment. As shown, output selector 500 includes control logic 510 and memory 511. In one embodiment, programming signals 504 are received, and in response, one or more data structures in memory 511 are updated.

FIG. 5B illustrates one data structure used in one embodiment. Available array 520 is programmed with an associative memory blocks and optionally previous stage results available for use indicator 525 for each profile ID 521 to be used. Each indicator 525 identifies which, if any, associative memory blocks, sets of entries or associative memory banks are to be considered in determining which matching associative entry to select for the ultimate highest-priority matching associative memory entry. In one embodiment, indicator 525 further identifies which previous stage results to consider. In one embodiment, a priority level is associated with each of the banks and/or previous stage results. Thus, based on a profile ID 521 received over via selector control signal 501 (FIG. 5A), available array 520 can be retrieved from memory 511 (FIG. 5A). In one embodiment, there is an implied priority ordering of associative memory blocks and any previous stage results, while in one embodiment this priority ordering for determining the ultimate highest-priority matching entry is programmable and/or variable per lookup operation. In one embodiment, associative memory blocks available for use indicator 525 is a bitmap data structure, while in one embodiment, associative memory blocks available for use indicator 525 is a list, set, array, or any other data structure. *Col 17, line 11*

Returning to FIG. 5A, in the performance of a lookup operation, output selector 500 receives selector control signal 501, which may include a profile ID. In addition, output selector 500 receives any relevant previous stage results 502 and results 503 from zero or more of the associative memory blocks from which the highest-priority entry will be selected, and which, if any, will be identified in generated result 515.



FIG. 9F illustrates a process used by one embodiment to perform lookup operations and to identify the merged result. Processing begins with process block 950, and proceeds to process block 951, wherein a lookup value is identified. Next, in process block 952, lookup operations are performed in the first and second associative memory banks and adjunct memories to generate first and second lookup results, which are merged in process block 953 to identify the merged result. Processing is complete as indicated by process block 954.

FIG. 9G illustrates a lookup value 960, result value 965, and merged result value 967 used in one embodiment. As shown, lookup value 960 includes a lookup type 960A, source address 960B, destination address 960C, source port 960D, destination port 960E, protocol type 960F, source ASN 960G, destination ASN 960H, and possibly other fields 960I. One embodiment uses all, less than all, or none of fields 960A-960I.

As shown, result value 965 includes a result type 965A, an action or counter indication 965B, and a precedence indication 965C. In one embodiment, result value 965 is programmed in the adjunct memories. One embodiment uses all, less than all, or none of fields 965A-965C.

As shown, merged result value 967 includes a result type 967A and an action or counter indication 967B. One embodiment uses all, less than all, or none of fields 967A-967B.

FIGs. 9H-9J illustrate merging logic truth tables 970, 972, and 974 for generating the merged result. In one embodiment, the merge result of a security lookup operation is illustrated in security combiner logic 970, and is based on the results of up to four substantially simultaneous (or not) lookup operations with differing precedence indicated in columns 970A-970D, with the corresponding merged result shown in column 970E.

Note, the "---" in the fields indicate a don't care condition as a merged result corresponding to a higher priority will be selected.

In one embodiment, the merge result of a Quality of Service (QoS) lookup operation is illustrated in security combiner logic 972, and is based on the results of a